Journal of Characterization cilt 4, sayı 1, Mart, pp 29-44, 2024



Araştırma Makalesi

Journal of Characterization

www.jcharacterization.org

ISSN: 2757-9166_Copyright © 2021 JCHAR



Instagram Gerçek ve Sahte Kullanıcıların Analizi

Engin Oğuzay

Department of Computer Programming, Maltepe University, Istanbul, Turkiye (ORCID: (0000-0003-2030-2981), enginoguzay@maltepe.edu.tr

(İlk geliş tarihi: 29.12.2023, Düzenleme tarihi: 20.03.2024 ve Kabul tarihi: 22.03.2024)

(**DOI: 10.29228/JCHAR.74443**)

Sorumlu Yazar : Engin Oğuzay, Department of Computer Programming, Maltepe University, Istanbul, Turkiye, enginoguzay@maltepe.edu.tr

Referans gösterme : E. Oğuzay, "Instagram Gerçek ve Sahte Kullanıcıların Analizi", *J Characterization*, cilt. 4, sayı. 1, Mart, 29-44, 2024, doi:10.29228/JCHAR.74443

Öz

Bu makalede; Instagram'daki sahte hesapların ve bu hesaplar üzerinden dolandırıcılık sorunlarının son yıllarda fazlasıyla artması sebebiyle, kullanıcı hesaplarının gerçek veya sahte olma olasılıklarını sınıflandırmak amacıyla dört farklı veri madenciliği algoritmasının kullanımını detaylı bir şekilde ele almaktadır. Her bir algoritmanın temel prensipleri, avantajları ve uygulama süreçleri incelenerek, Instagram kullanıcı davranışlarını analiz etmek ve sahte hesapları tespit etmek için nasıl kullanılabilecekleri açıklanmıştır. Makalede, projenin ilk aşamasında 60.000'den fazla veri içeren zengin bir veri seti kullanılarak her bir algoritmanın bu veri seti üzerinde nasıl uygulandığı ve sonuçlarının nasıl değerlendirildiği detaylandırılmıştır. Özellikle, model performanslarını ölçmek için kullanılan AUC, Accuracy, Mean Per-Class Error, LogLoss ve Confusion Matrix gibi metrikler üzerinde durulmuştur. Makalenin sonucunda, dört modelin karşılaştırılması yapılarak, her modelin güçlü ve zayıf yönleri tartışılarak, Instagram hesaplarının sınıflandırılmasında hangi algoritmanın daha etkili olduğuna dair değerli bilgiler sunulmuştur. Bu bilgiler, kullanıcı deneyiminin iyileştirilmesi ve platform güvenliğinin artırılması gibi alanlarda kullanılabilir.

Anahtar Kelimeler : Algoritma, Instagram, Modelleme, RapidMiner, Veri Madenciliği

Analysis of Instagram Real and Fake Users

Abstract

In this article, due to the increasing number of fake accounts on Instagram and the fraud problems associated with these accounts in recent years, it discusses in detail the use of four different data mining algorithms to classify the likelihood of user accounts on Instagram being real or fake. The basic principles, advantages, and application

processes of each algorithm are examined, and how they can be used to analyze Instagram user behavior and detect fake accounts is explained. In the article, a rich dataset containing more than 60,000 data points was used in the first stage of the project. It details how each algorithm was applied to this dataset and how the results were evaluated. In particular, metrics such as AUC, Accuracy, Mean Per-Class Error, LogLoss, and Confusion Matrix, which are used to measure model performance, are emphasized. As a result of the article, valuable information is provided about which algorithm is more effective in classifying Instagram accounts by comparing four models and discussing the strengths and weaknesses of each model. This information can be used in areas such as improving user experience and increasing platform security.

Keywords: Algorithm, Instagram, Modeling, RapidMiner, Data Mining

1. Giriş

Günümüz dijital çağında, sosyal medya platformları sadece iletişim ve etkileşim araçları olmanın ötesine geçmiş, kullanıcı davranışlarının, trendlerin ve hatta sahte hesapların incelenmesi için verimli bir alan haline gelmiştir. Özellikle Instagram, görsel ağırlıklı yapısı ve geniş kullanıcı kitlesiyle, veri madenciliği için ideal bir ortam sunmaktadır. Bu çalışmada, Instagram'da bulunan kullanıcı hesaplarının gerçek ya da sahte olma olasılıklarını sınıflandırma amacıyla dört farklı veri madenciliği algoritmasının kullanıldığı bir analiz sürecini ele alınmıştır. İncelenen algoritmalar; Deep Learning, Karar Ağacı, Lojistik Regresyon ve Naive Bayes'tir. Her bir algoritmanın temel prensipleri, avantajları ve bu spesifik çalışmada nasıl uygulandıkları detaylandırılacaktır.

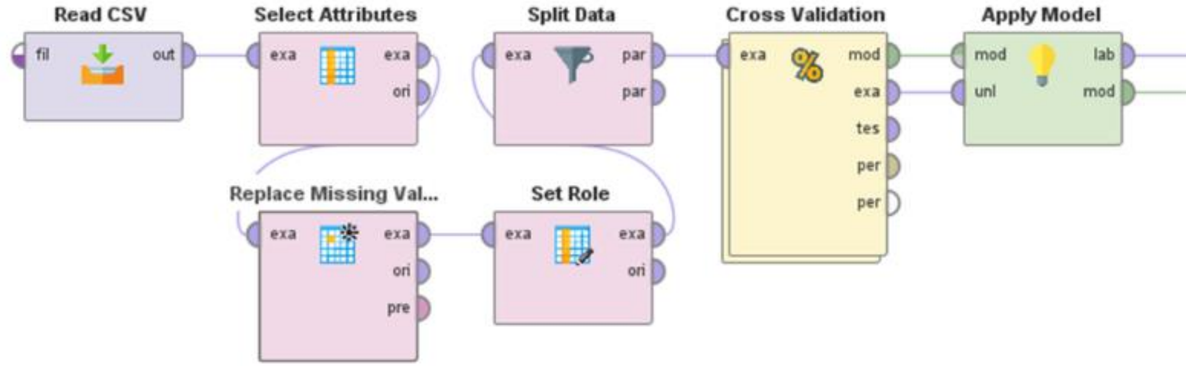
Makalenin önemi, sahte kullanıcıların tespit edilmesinin, reklam verimliliğinden kullanıcı güvenliğine kadar pek çok alanda etkili olmasıdır. Özellikle markalar için sahte takipçi ve etkileşim oranlarının doğru bir şekilde analiz edilmesi, pazarlama stratejilerinin başarısını doğrudan etkileyebilmektedir. Ayrıca, sahte hesapların tespiti, kullanıcı deneyimini iyileştirme ve platform güvenliğini artırma açısından da büyük öneme sahiptir.

Bu analiz sürecinde kullanılan RapidMiner yazılımı, veri madenciliği ve makine öğrenimi projeleri için güçlü ve esnek bir araçtır. Kullanıcı dostu arayüzü, geniş algoritma kütüphanesi ve veri işleme kabiliyetleri ile hem yeni başlayanlar hem de deneyimli analistler için idealdir. RapidMiner'in bu çalışmada nasıl kullanıldığına dair teknik detaylar, ilerleyen bölümlerde daha ayrıntılı bir şekilde incelenecektir.

2. Materyal ve Metod

2.1. Derin Öğrenme Algoritması

Derin Öğrenme (Deep Learning) Algoritması, Yapay Zeka (Artificial Intelligence – AI) ve makine öğrenmesinin en ileri ve etkileyici alanlarından biridir. Bu algoritma, insan beyninin bilgi işleme biçimini taklit eden yapay sinir ağları (ANN) üzerine kurulmuştur. Derin Öğrenmenin temel amacı, makinelere veri üzerinden öğrenme ve bu veriler içerisindeki karmaşık örüntüleri ve ilişkileri tanıma yeteneği kazandırmaktır. Aşağıdaki Şekil 1’de Derin Öğrenme Algoritması Uygulaması verilmiştir.



Şekil 1 : Derin Öğrenme Algoritması Uygulaması

1. Veri Setinin Hazırlanması

Çalışmada, 10.000 veri noktasını içeren zengin bir veri seti kullanılmıştır. Bu büyük veri seti, algoritmanın karmaşık örüntüleri tanıma ve genelleme yapma kabiliyetini artırır. Büyük veri setleri, modelin daha doğru ve güvenilir sonuçlar üretmesini sağlar.

2. Öznitelik Seçimi

"Select Attributes" adımında, algoritmanın kullanacağı özellikleri (attributes) seçtim. Bu seçim, veri setindeki en önemli ve etkili özellikleri belirlemek için kritik bir adımdır. İrrelevant veya yanıltıcı özelliklerin çıkarılması, modelin performansını artırır ve eğitim süresini kısaltır.

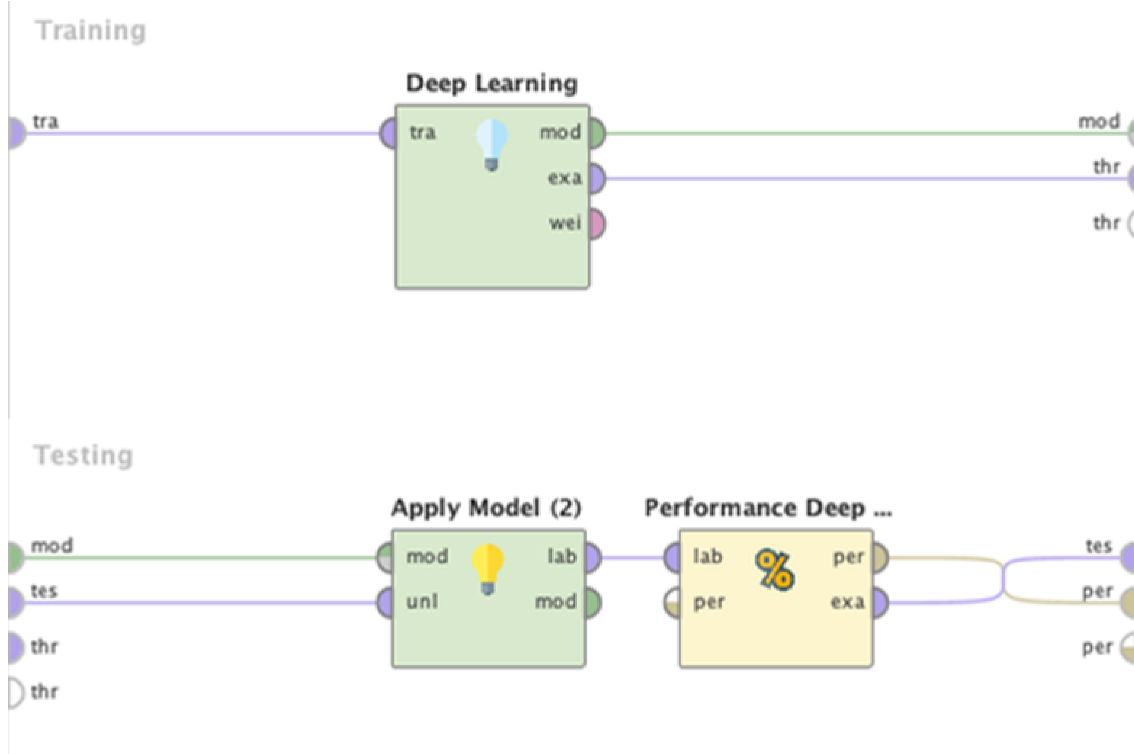
3. Rol Ataması

"Setrole" işlemiyle, "sınıf" özelliğine "label" rolü verilmiştir. Bu adım, algoritmanın hangi özelliğin tahmin edilecek hedef (target) olduğunu anlamasını sağlar. Bir Deep Learning modelinde, doğru etiketlemenin yapılması, modelin neyi öğrenmesi gerektiğini belirler.

4. Veri Bölümleme

"Split Data" işlemiyle verileri %70 gerçek (eğitim) ve %30 test verisi olarak ayrılmıştır. Bu bölümleme, modelin eğitilmesi ve test edilmesi için gerekli olan veri yapılandırmasını sağlar. Eğitim verileri modeli eğitmek için kullanılırken, test verileri modelin gerçek dünya verileri üzerindeki performansını değerlendirmek için kullanılır.

5. Çapraz Doğrulama ve Model Uygulaması
"Cross Validation" adımında, Deep Learning modeli kullanılmıştır. Bu süreçte "Apply Model" uygulayarak modelin performansını (classification) değerlendirilmiş olup, çapraz doğrulama, modelin farklı veri alt kümeleri üzerindeki performansını test eder, böylece modelin genelleştirme yeteneği hakkında daha sağlam bir fikir edinilir. Aşağıdaki Şekil 2'de Derin Öğrenme Algoritması Çapraz Doğrulama İşlemi gösterilmektedir.



Şekil 2 : Derin Öğrenme Algoritması, Çapraz Doğrulama İşlemi

6. Son Model Uygulaması

Son olarak, tüm adımları "Apply Model" ile tamamlanmıştır. Bu adım, eğitilmiş modelin tüm veri seti üzerinde nasıl performans gösterdiğini görmemi sağlamıştır.

Modelin bizim için vermiş olduğu sonuçlardan önemli olanları inceleyecek olursak sonuçlar aşağıdaki şekilde:

AUC (Area Under the Curve): Modelin AUC değeri 0.85 olarak ölçülmüştür. Bu, ROC eğrisi altında kalan alanın büyüklüğünü ifade eder ve modelin sınıflandırma kapasitesinin oldukça iyi olduğunu gösterir. AUC değeri 1'e yakın olduğunda, modelin sınıflandırma yeteneği daha yüksek olarak değerlendirilir.
Accuracy: Modelin doğruluğu %93.87 olarak belirlenmiştir. Bu oran, modelin verilerin büyük bir kısmını başarıyla sınıflandırdığını gösterir ve genel performansın yüksek olduğuna işaret eder.

Mean Per-Class Error: Bu oran, sınıfların ortalama olarak yaklaşık %10'unun yanlış sınıflandırıldığını gösterir. Bu, modelin her sınıf üzerindeki performansının dengeli olduğu anlamına gelir.

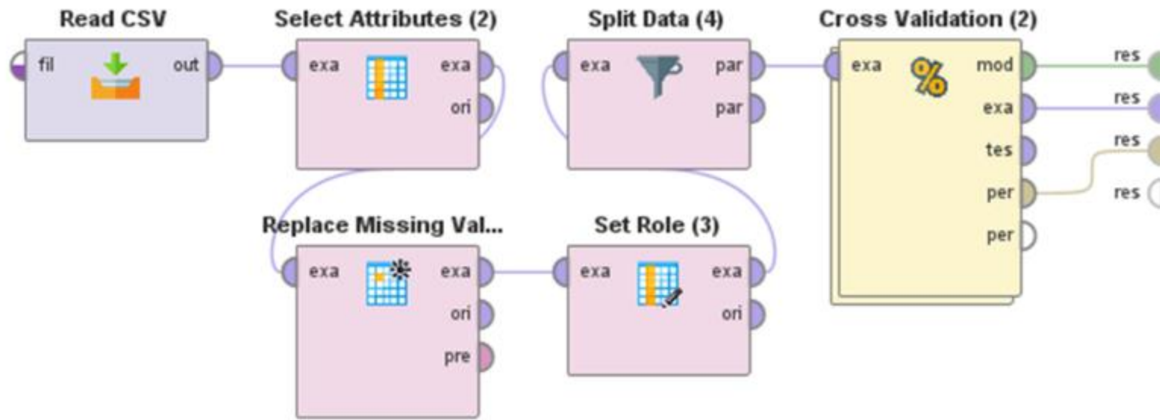
LogLoss: LogLoss, modelin tahminlerinin güvenilirliğini ölçer. Düşük bir LogLoss değeri, modelin iyi kalibre edildiğini ve güvenilir tahminler yaptığını gösterir. sonuç olarak 0.2'lik bir değer olduğunu ve modelin yeterince güvenilir bir sonuç verdiğini gösterir.

Confusion Matrix: Modelin, doğru pozitif (3858), yanlış negatif (1023), doğru negatif (2020) ve yanlış pozitif (1) tahminlerini içerir. Bu dağılım, modelin pozitif sınıfı tespit etme konusunda güçlü olduğunu, ancak bazı negatif durumları kaçırdığını gösterir.

Genel bir toparlayacak olarsak; modelin bize sunmuş olduğu sonuçlara baktığımızda iyi bir sınıflandırma yeteneğine işaret ediyor. %93.87 doğruluk ve 0.85 AUC değeri, modelin yüksek bir sınıflandırma başarısına sahip olduğunu göstermektedir.

2.2. Karar Ağacı Algoritması

Karar Ağacı; veri madenciliği ve makine öğrenmesinde kullanılan popüler bir sınıflandırma ve regresyon yöntemidir. Bu algoritma, karar verme süreçlerini taklit ederek, veri setini daha küçük alt kümelere ayırır ve sonuçta bir 'ağaç' yapısı oluşturur. Genel yapısı karar düğümleri ve sonuç dallarından oluşur. Her düğüm, bir özelliğin değerini test eder ve dallar, bu testin sonuçlarına göre ayrılır. Kolay anlaşılabilir ve görselleştirilmesi basit olması avantajlarındandır. Ancak, aşırı öğrenme (overfitting) riski taşıyabilir ve bazen veri değişikliklerine karşı hassas olabilir. Aşağıdaki Şekil 3'de Karar Ağacı Algoritma Uygulaması verilmiştir.



Şekil 3 : Karar Ağacı Algoritması, Uygulama

Bu algoritmayı kullanırken, performans sonuçlarını karşılaştırmak için yine Deep Learning Algoritmasında kullanmış olduğum veri seti kullanılmıştır ancak bazı kayıp veriler olabileceğinden "Replace Missing Value" özelliğini uygulamaya eklenerek devam edilmiştir.

Tablo 1 : Karar Ağacı Algoritması, Performans Sonucu

Performance Vector		
Accuracy	%87.33 ±0.50% (micro average: 87.33%)	
Confusion Matrix		
True:	Fake (f)	Real (r)
Fake (f)	17543	333
Real (r)	5463	22389

Accuracy: Modelin doğruluğu %87.33 ±0.5 olarak verilmiş. Bu değer, modelin tahminlerinin genel olarak yüksek bir doğrulukla doğru olduğunu gösterir.

Karışıklık Matrisi:

True f (Gerçek Negatif): 17543 - Modelin negatif sınıfı (f) olarak doğru tahmin ettiği örnek sayısı.

False r (Yanlış Pozitif): 333 - Modelin aslında negatif olan (f) örnekleri pozitif (r) olarak yanlış tahmin ettiği örnek sayısı.
False f (Yanlış Negatif): 5463 - Modelin aslında pozitif olan (r) örnekleri negatif (f) olarak yanlış tahmin ettiği örnek sayısı.
True r (Gerçek Pozitif): 22389 - Modelin pozitif sınıfı (r) olarak doğru tahmin ettiği örnek sayısı.

Bu matrise göre model, pozitif sınıfı (r) oldukça iyi tanımış görünüyor. Ancak, negatif sınıfı (f) için yanlış pozitif oranı, modelin bazı durumlarda negatif örnekleri yanlışlıkla pozitif olarak sınıflandırdığını gösteriyor. Genel olarak, modelin yüksek bir doğruluk oranına sahip olduğunu ve özellikle pozitif sınıfı iyi tanıdığını söyleyebiliriz.

Karar ağacı, verilen özellikler üzerinden bir dizi karar kuralı kullanarak sınıflandırma yapar. Paylaşılan karar ağacı kuralları, modelin nasıl tahminler yaptığını gösteriyor. Her kural, belirli bir özelliğin değerine göre dallara ayrılıyor ve bir sınıf tahmini (f veya r) ile sonuçlanıyor.

Örneğin, ilk kurala göre takipEdilen > 8050 ise sınıf r. Bu, takip edilen sayısı 8050'den büyük olan örneklerin r sınıfına ait olduğunu gösteriyor. Diğer kurallar da benzer şekilde, özelliğin değerine göre dal ayrımı yaparak sınıflandırma yapıyor.

Karar ağacı kuralları, modelin hangi özelliklere göre hangi sınıflandırmaları yaptığını anlamak için faydalıdır. Bu, modelin anlaşılabilirliği ve açıklanabilirliği açısından önemlidir. Ayrıca, bu kurallar üzerinden modelin nerede iyi performans gösterdiğini ve hangi özelliklerin önemli olduğunu görebiliriz.

Genel bir toparlayacak olursak, modelin genel olarak yüksek bir doğruluk oranına sahip olduğu ve pozitif sınıfı iyi tanıdığı söylenebilir. Ancak, her modelin gerçek dünya senaryolarında da test edilmesi gerekir. Modelin performansını iyileştirmek için yanlış pozitifleri ve yanlış negatifleri azaltmaya yönelik daha fazla çalışma yapılabilir.

2.3. Bagging ile Karar Ağacı Algoritması

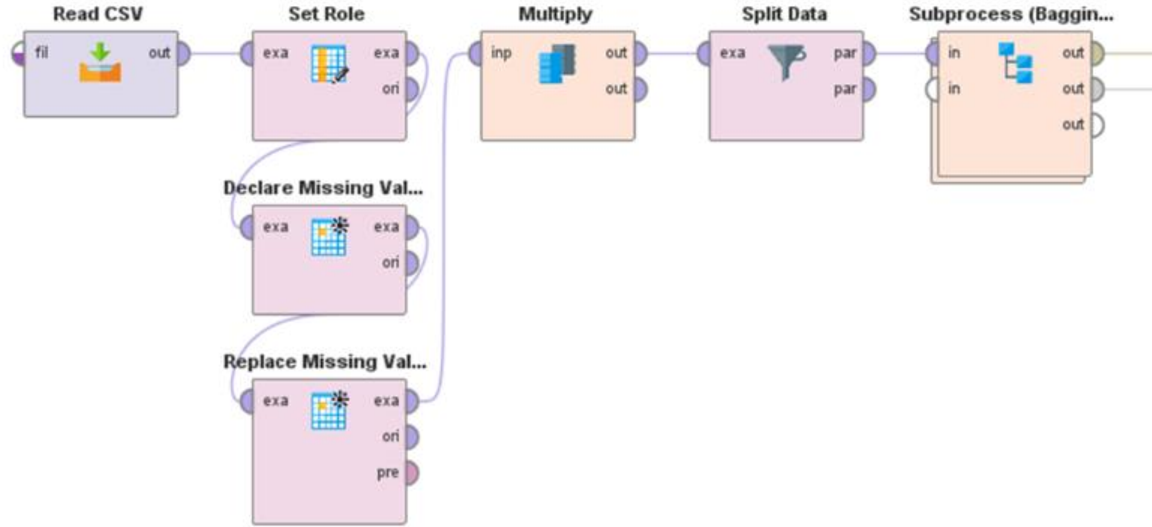
Bagging (Bootstrap Aggregating) yöntemiyle oluşturulan karar ağaçları topluluğunun avantajları ve işleyiş mekanizmasını kullanarak, karar ağacı algoritmasının vermiş olduğu performans ve hata oranı değerlerini iyileştirmesi amacıyla incelenmiştir. Bagging, bir topluluk öğrenme tekniği olarak, birden fazla karar ağacının tahminlerini birleştirerek, tek bir karar ağacına kıyasla daha güçlü ve genel geçer bir model oluşturmayı amaçlar.

Örnek Çeşitliliği ve Güçlü Öğrenme: Bagging yöntemi, veri setinin rastgele alt kümelerini kullanarak birçok karar ağacı oluşturur. Her ağaç, farklı veri örneklerinden öğrenir, bu da her birinin veri setinin farklı yönlerine odaklanmasını sağlar. Bu yöntem, çeşitli örneklemelerle elde edilen geniş bir bakış açısı sunar.

Aşırı Uyuma Karşı Direnç: Karar ağaçları, özellikle küçük ve özgül veri setlerinde aşırı uyuma (overfitting) eğilimindedir. Bagging, bu sorunu azaltarak genel tahmin yeteneğini artırır. Her ağacın bağımsız olarak eğitilmesi ve sonrasında birleştirilmesi, bireysel ağaçların zayıflıklarını dengeler.

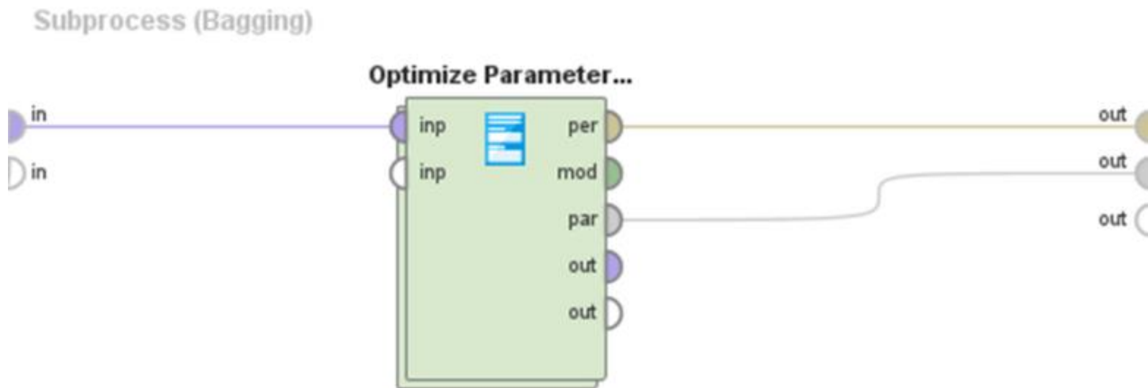
Daha Dengeli ve Güvenilir Tahminler: Bagging, oluşturulan her ağacın tahminlerini birleştirerek nihai bir karar verir. Bu süreç, bireysel ağaçların yapabileceği hataları minimize eder ve daha istikrarlı sonuçlar üretir.

Modelin Genelleştirme Kapasitesinin Artması: Bagging yöntemi, modelin küçük veri seti değişikliklerine karşı daha dayanıklı olmasını sağlar. Bu, genel olarak modelin daha esnek ve genelleştirilebilir olmasına katkıda bulunur. Aşağıdaki Şekil 4'de Bagging ile Karar Ağacı Algoritma Uygulaması verilmiştir.



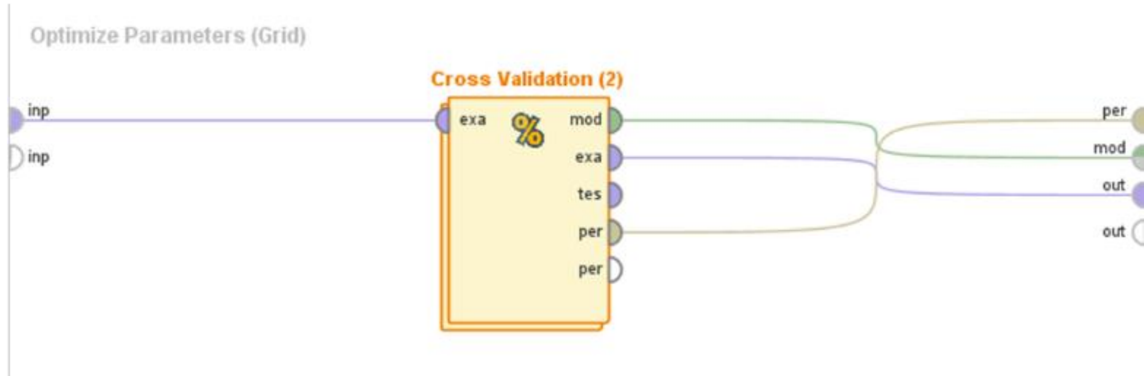
Şekil 4 : Bagging ile Karar Ağacı Algoritması, Uygulama

Karar ağacı algoritmasının doğruluk ve hata sapma oranını iyileştirmek amacıyla öncelikle veri seti üzerinde verileri daha detaylı işledikten sonra sonuçları karşılaştırmak amacıyla bir multiply işleminde incelenmiş ve bu şekilde iki algoritmanın sonuçları incelenmiştir. Subprocess içerisinde bagging meta-algoritması kullanılmıştır. Aşağıdaki Şekil 5'de Bagging ile Karar Ağacı Algoritması, Subprocess altında parametrelerin optimizasyonu işlemi verilmiştir.



Şekil 5 : Bagging ile Karar Ağacı Algoritması, Subprocess altında parametrelerin optimizasyonu işlemi

Optimize Parameters (Grid) operatörü, belirli bir model için en iyi parametre değerlerini bulmak amacıyla kullanılır. Bu operatör, belirtilen parametreler için bir grid (ızgara) araması yapar. Izgara araması, aday parametre değerlerinin her bir kombinasyonu için modeli eğitir ve bu kombinasyonların performansını değerlendirir. Aşağıdaki Şekil 6'da Bagging ile Karar Ağacı Algoritması, Optimize Parameter (Grid) altında Cross Validation Kullanılması işlemi verilmiştir.



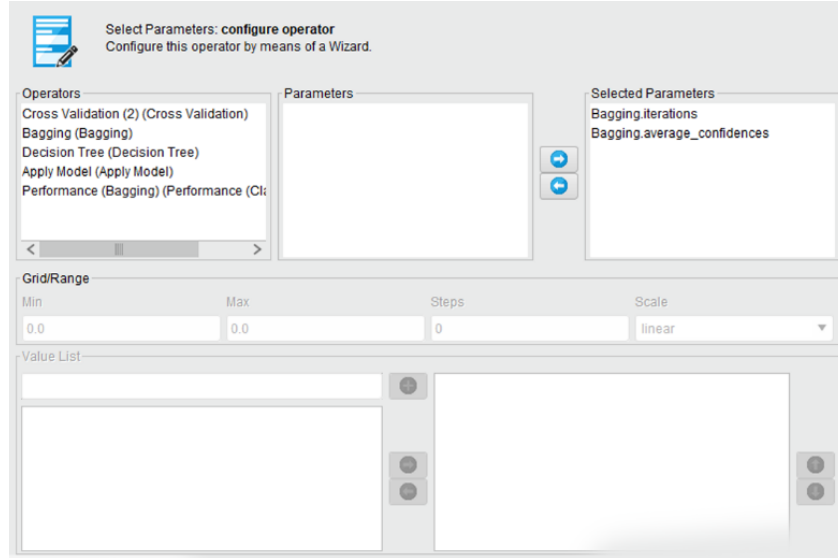
Şekil 6 : Bagging ile Karar Ağacı Algoritması, Optimize Parameter (Grid) altında Cross Validation Kullanılması

Bagging algoritması parametre ayarlarını sample ratio(0.7), iteration(10) olarak ayarlanmıştır. Bagging algoritması, orijinal eğitim veri setinden rastgele alt kümeler (bootstrap örnekleri) çekmek için kullanılır. "Sample ratio", her bir bootstrap örneğinin orijinal veri setine oranını belirtir. Eğer bu oran 0.9 olarak ayarlanmışsa, bu, her bir alt kümenin orijinal veri setinin %90'ını içereceği anlamına gelir. Bu, her bir bagging iterasyonunda kullanılacak veri miktarını kontrol eder. Iterations, bağımsız modellerin kaç kez eğitileceğini belirtir. Eğer bu değer 10 olarak ayarlanmışsa, bagging algoritması 10 farklı model eğitecek ve her bir model farklı bir bootstrap örneğini kullanacaktır. Her bir iterasyon sonucunda elde edilen modellerin tahminleri daha sonra birleştirilir (örneğin, çoğunluk oyu veya ortalama alma yoluyla) ve nihai bir tahmin elde edilir. Aşağıdaki Şekil 7'de Bugging algoritması parametre ayarları işlemi verilmiştir.

Bagging
sample ratio 0.9
iterations 10
average confidences
use local random seed

Şekil 7 : Bagging algoritması parametre ayarları

Tüm işlemleri hallettikten sonra Optimization parameters(Grid) işlemimize geri dönüp ayarlarından modelin performansını etkilediği düşünülen parametreleri seçerek her biri için bir değer aralığı belirlenmiştir. Seçilen parametrelerin her bir kombinasyonu için bir ızgara oluşturulmuş. bu her bir parametre kombinasyonu için model eğitilmiş ve belirli bir performans ölçütüne göre değerlendirilmiştir. Bu, doğruluk, RMSE veya başka bir metrik olabilir. Aşağıdaki Şekil 8'de Optimization Parameter(Grid), parametre ayarları işlemi verilmiştir.



Şekil 8 : Optimization Parameter(Grid), parametre ayarları

Tablo 2: Bagging ile Karar Ağacı Algoritması, Performans (Normal) Sonuçları

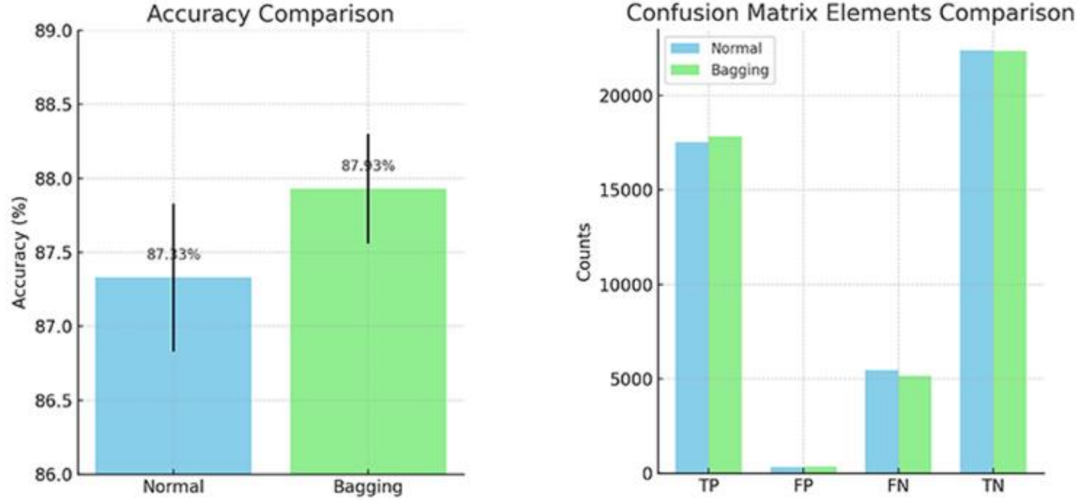
Performance Vector		
Accuracy	%87.33 ±0.50% (micro average: 87.33%)	
Confusion Matrix		
True:	Fake (f)	Real (r)
Fake (f)	17543	333
Real (r)	5463	22389

Yapılan testler ve analizler sonucunda, bagging yöntemiyle geliştirilen karar ağacı modelinin, geleneksel karar ağacı modeline göre %0.6 oranında performans iyileştirmesi sağladığı ve hata oranını 0.3 ile birbirine çok yakın

Tablo 3 : Bagging ile Karar Ağacı Algoritması, Performans (Bagging) Sonuçları

Performance Vector		
Accuracy	%87.93 ±0.37% (micro average: 87.93%)	
Confusion Matrix		
True:	Fake (f)	Real (r)
Fake (f)	17843	357
Real (r)	5163	22365
Root Mean Squared Error	0.327 +/- 0.005 (micro average: 0.327 +/- 0.000)	

olduğu gözlemlenmiştir. Bu, bagging yönteminin, karar ağacı algoritmalarını daha etkin ve güvenilir hale getirmede ne kadar başarılı olduğunun bir göstergesidir. Aşağıdaki Şekil 9'da Modellerin Karşılaştırma Sonuçları gösterilmektedir

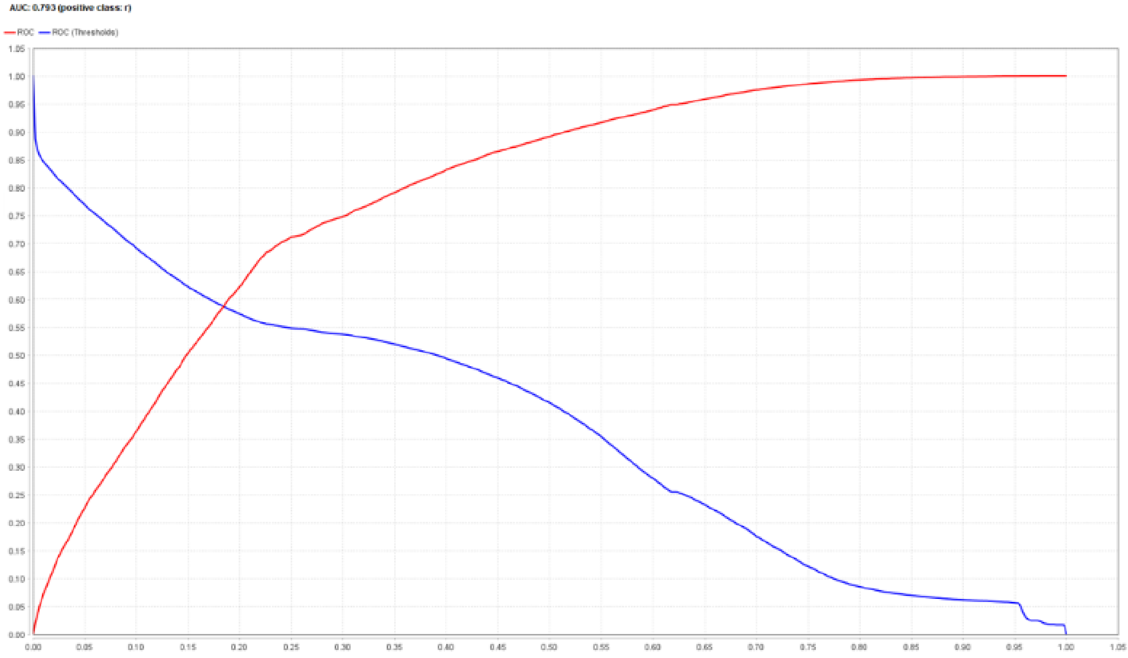


Şekil 9 : Modellerin Karşılaştırma Sonuçları

2.4. Lojistik Regresyon Algoritması

Lojistik Regresyon, özellikle ikili sınıflandırma problemleri için kullanılan bir istatistiksel modelleme tekniğidir. Bu yöntem, bağımsız değişkenlerin bir sonucu (genellikle 0 veya 1) nasıl etkilediğini modellemek için kullanılır. Mekanizması veri noktalarının belirli bir kategoriye ait olma olasılıklarını tahmin eder. Sonuç, genellikle bir sigmoit fonksiyonu kullanılarak 0 ile 1 arasında bir değer olarak verilir. Basit ve verimli olması en büyük avantajlarından biridir. Ancak, doğrusal ilişkileri varsayar ve karmaşık ilişkileri modellemede sınırlı olabilir.

Bir diğer algoritmayı uygularken aynı veri setini kullanmaya devam edilmiş ve çapraz doğrulama işlemine sokmadan algoritmanın kendisi kullanılmıştır. Aşağıdaki Şekil 10'da Lojistik Regresyon Algoritması AUC, ROC Grafik Sonucu gösterilmektedir.



Şekil 10 : Lojistik Regresyon Algoritması, AUC, ROC Grafik Sonucu

ROC eğrisi, modelin farklı eşik değerlerindeki gerçek pozitif oranı (TPR) ile yanlış pozitif oranı (FPR) arasındaki ilişkiyi gösterir. Eğri ne kadar yukarı ve sola yakınsa, model o kadar iyi performans gösterir. AUC (Area Under the Curve), eğrinin altındaki alanın büyüklüğünü ifade eder ve ideal bir sınıflandırıcı için 1'e eşittir. Gönderdiğiniz ROC eğrisinde AUC 0.793 olarak belirtilmiş, bu değer modelin oldukça iyi bir sınıflandırma performansına sahip olduğunu gösterir ancak mükemmel olmadığını da belirtir.

Tablo 4 : Lojistik Regresyon Algoritması, Performans Sonucu

Performance Vector		
Accuracy	%71.63	
Confusion Matrix		
True:	Fake (f)	Real (r)
Fake (f)	14040	4006
Real (r)	8966	18716

Accuracy (Doğruluk): %71.63, bu modelin genel olarak %71.63 doğru tahmin yaptığı anlamına gelir.

Karışıklık Matrisi:
True Negative (TN) - f f: 14040 - Modelin negatif sınıfı doğru tahmin ettiği örnek sayısı.
False Positive (FP) - f r: 4006 - Modelin aslında negatif olan örnekleri yanlışlıkla pozitif olarak sınıflandırdığı örnek sayısı.
True Positive (TP) - r r: 18716 - Modelin pozitif sınıfı doğru tahmin ettiği örnek sayısı.

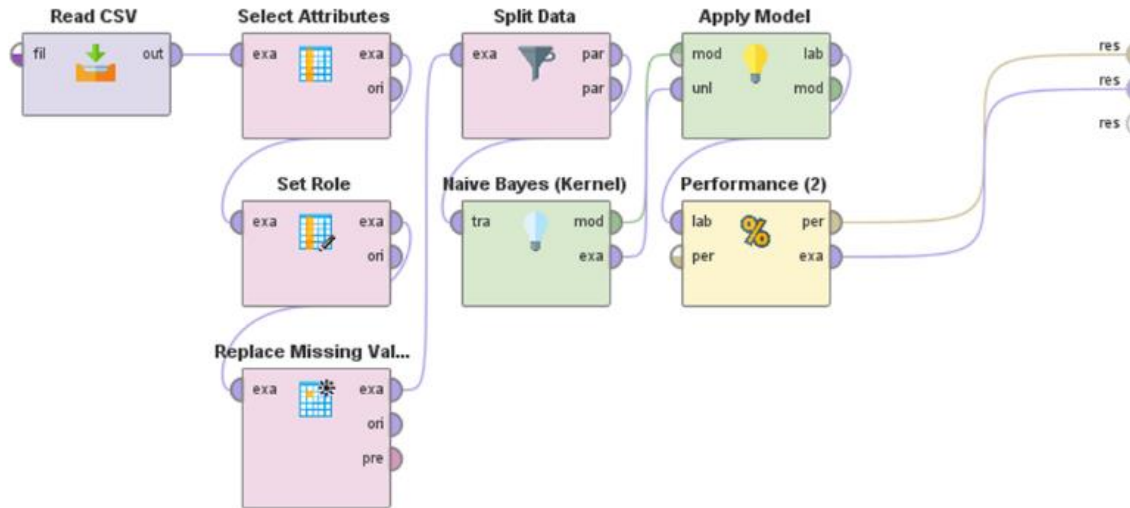
False Negative (FN) - r f: 8966 - Modelin aslında pozitif olan örnekleri yanlışlıkla negatif olarak sınıflandırdığı örnek sayısı.

Modelin doğruluğu makul bir seviyede olsa da, özellikle yanlış negatiflerin (r f) ve yanlış pozitiflerin (f r) sayısının yüksek olması, bazı durumlarda modelin performansını iyileştirmek için çalışma yapılması gerektiğini gösterir.

Genel sonuca bakacak olursak; modelin AUC değeri, genel olarak iyi bir performans sergilediğini gösteriyor, ancak doğruluğu %71.63 ile idealin altında kalıyor. Bu durum, modelin bazı durumlarda hatalı sınıflandırmalar yaptığını ve belirli sınıflarda iyileştirme yapılması gerektiğini gösteriyor. Modelin doğruluğunu ve AUC değerini artırmak için daha fazla özellik mühendisliği, hiperparametre optimizasyonu ve belki de veri setindeki dengesizlikleri düzeltme gibi stratejiler uygulanabilir. Yanlış pozitif ve yanlış negatif oranlarını azaltmak için eşik değerinin ayarlanması da düşünülebilir.

2.5. Naive Bayes Algoritması

Naive Bayes, olasılık temelli bir sınıflandırma yöntemidir ve adını, İngiliz istatistikçi Thomas Bayes'in adını taşıyan Bayes Teoremi'nden alır. Bu algoritma, verilen bir veri noktasının hangi sınıfa ait olduğunu tahmin etmek için kullanılır. Genel işleyişi, her özelliğin sınıf üzerindeki etkisinin diğer özelliklerden bağımsız olduğu varsayımından gelir. Yani, bir özelliğin varlığı, diğer özelliklerin varlığını veya yokluğunu etkilemez. Bu basitleştirme, algoritmanın hesaplama süresini önemli ölçüde azaltır ve uygulanmasını kolaylaştırır. Aşağıdaki Şekil 11'de Naive Bayes Algoritma Uygulaması gösterilmektedir. En büyük avantajlarından biri, basit ve hızlı olmasıdır. Küçük veri setlerinde bile iyi performans gösterebilir. Ancak, gerçek dünya verilerinde özelliklerin her zaman bağımsız olmaması nedeniyle, bazı durumlarda yanıltıcı sonuçlar verebilir.



Şekil 11 : Naive Bayes Algoritması, Uygulama

En son algoritmamız naive bayes'i aynı veri setini kullanmaya devam edilmiş, kayıp veriler olabilmesi durumunda "Replace Missing Value" kullanılmış ve çapraz doğrulama işlemine sokmadan algoritmanın kendisini kullanarak devam edilmiştir.

Tablo 5 : Naive Bayes Algoritması, Performans Sonuçları

Performance Vector		
Accuracy	%85.74	
Confusion Matrix		
True:	Fake (f)	Real (r)
Fake (f)	8270	1204
Real (r)	1590	8534
Classification Error	%14.26	
Absolute Error	0.178 +/- 0.266	
Relative Error	%17.83 +/- %26.63	
Normalized Absolute Error	354	
Root Mean Squared Error	0.320 +/- 0.000	
Root Relative Squared Error	637	
Squared Error	0.103 +/- 0.218	

Accuracy (Doğruluk): %85.74 - Modelin genel doğruluk oranı oldukça yüksek. Bu, modelin %85.74 oranında doğru tahminde bulunduğu anlamına gelir.

Classification Error (Sınıflandırma Hatası): %14.26 - Modelin genel hata oranı. Bu oran modelin yanlış tahmin yapma oranını ifade eder.

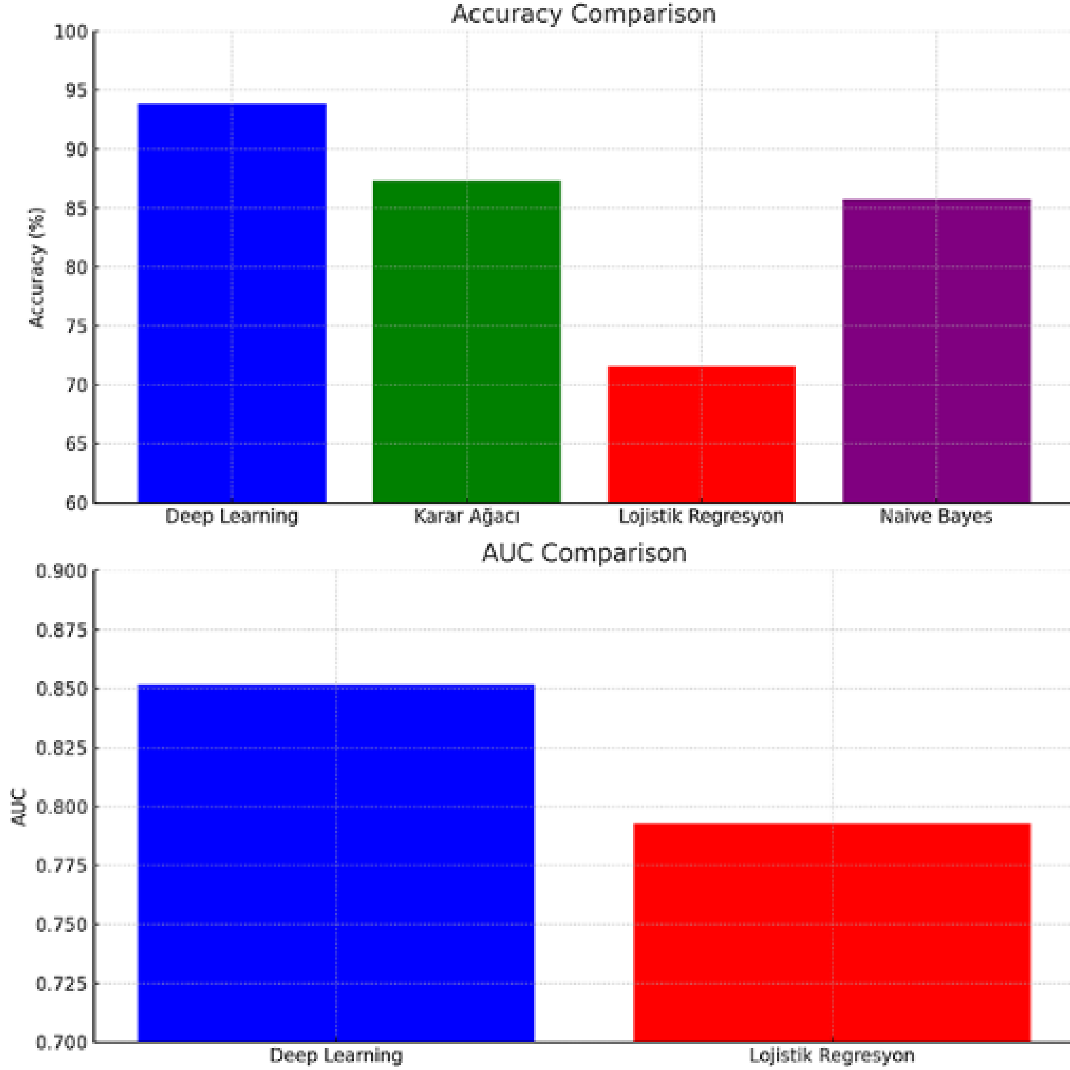
True Negative (TN) - f f: 8270 - Modelin negatif sınıfı (f) olarak doğru tahmin ettiği örnek sayısı.
False Positive (FP) - f r: 1204 - Modelin aslında negatif olan örnekleri yanlışlıkla pozitif (r) olarak sınıflandırdığı örnek sayısı.
False Negative (FN) - r f: 1590 - Modelin aslında pozitif olan örnekleri yanlışlıkla negatif (f) olarak sınıflandırdığı örnek sayısı.
True Positive (TP) - r r: 8534 - Modelin pozitif sınıfı (r) olarak doğru tahmin ettiği örnek sayısı.

Root Mean Squared Error (RMSE): 0.320 ±0.000 - Tahminlerin gerçek değerlerden ne kadar sapma gösterdiğinin karekök ortalaması. Bu değer, modelin hata marjını gösterir ve düşük olması tercih edilir.

Genel bir değerlendirme yaparsak, modelin doğruluk oranı oldukça iyi (%85.74). Bu, Naive Bayes algoritmasının bu veri seti için etkili bir sınıflandırma yapabildiğini gösterir. Ancak, karışıklık matrisinde yer alan yanlış pozitiflerin (1204) ve yanlış negatiflerin (1590) sayısını azaltarak modelin performansını daha da arttırılabilir.

Mutlak hata ve RMSE değerleri düşük, bu da modelin tahminlerinin gerçek değerlere genellikle yakın olduğunu gösterir. Göreceli hata yüzdesi ve standart sapmasının geniş aralığı, bazı tahminlerde daha büyük hataların olabileceğine işaret ediyor olabilir. Bu nedenle, modeli daha da iyileştirmek için özellik seçimi, veri ön işleme veya parametreler üzerinde değişkilikler yapılabilir.

Dört modelin karşılaştırılması sonucunda, Deep Learning modelinin en yüksek doğruluk ve AUC değerleriyle en iyi performans gösteren model olduğu sonucuna varılmıştır. En iyi modeli seçerken, aşağıdaki Şekil 12'de de gösterildiği gibi her modelin doğruluk oranı ve AUC değeri i kritik metrikleri belirtilmektedir.



Şekil 12 : Modellerin Karşılaştırma Sonuçları

3. Sonuçlar

Bu makalede, Instagram kullanıcı hesaplarının gerçek veya sahte olma olasılıklarını sınıflandırmak için dört farklı veri madenciliği algoritması incelenmiştir: Her bir algoritmanın avantajları ve dezavantajları, yanı sıra potansiyel iyileştirme alanları üzerinde durulmuştur. Deep Learning, Karar Ağacı, Lojistik Regresyon ve Naive Bayes. Her bir algoritmanın performansı, doğruluk, AUC değeri, karışıklık matrisi ve diğer önemli metrikler kullanılarak değerlendirilmiştir.

Tablo 6 : Algoritmaların Performans Karşılaştırması

Algoritma	Doğruluk (%)	AUC
Deep Learning	93.87	0.8514
Karar Ağacı	87.33	-
Lojistik Regresyon	71.63	793
Naive Bayes	85.74	-

Deep Learning Modeli: Deep Learning modeli, dört model arasında en yüksek doğruluk oranına ve AUC değerine sahip. Bu, modelin genel olarak veri setindeki örnekleri doğru bir şekilde sınıflandırma konusunda diğerlerinden daha başarılı olduğunu gösterir. Ancak, Deep Learning modelleri genellikle daha fazla hesaplama gücü gerektirir ve yorumlanması daha zordur. Yüksek hesaplama gereksinimlerini azaltmak ve model yorumlanabilirliğini artırmak için farklı tekniklerin uygulanması gerekebilir. Ayrıca, yanlış negatif ve yanlış pozitif oranlarını düşürmek amacıyla hiperparametre ayarlamaları yapılabilir.

Karar Ağacı Modeli: Karar Ağacı modeli, yüksek bir doğruluk oranına sahip ve kurallarının açık olması nedeniyle yorumlanabilirliği yüksektir. Ancak, bu model, Deep Learning modeline göre daha düşük bir performansa sahip. Aşırı öğrenme (overfitting) riskini azaltmak ve veri değişikliklerine olan hassasiyetini düşürmek için düzenleme ve sınırlama teknikleri kullanılabilir.

Lojistik Regresyon Modeli: Lojistik Regresyon modeli, doğruluk ve AUC değerleri bakımından diğer modellere göre daha düşük performans gösteriyor. Ancak, modelin basitliği ve yüksek yorumlanabilirliği, belirli uygulamalar için tercih edilebilir. Modelin karmaşık ilişkileri daha iyi modelleyebilmesi için özellik mühendisliği ve hiperparametre optimizasyonu gibi yöntemler uygulanabilir.

Naive Bayes Modeli: Naive Bayes, iyi bir doğruluk oranına sahip ve hızlı eğitim süresi gibi avantajlara sahip olduğu için büyük veri setleri veya gerçek zamanlı uygulamalar için uygun olabilir. Özellik bağımsızlığı varsayımının etkilerini azaltmak ve yanlış pozitif/negatif oranlarını düşürmek için veri ön işleme ve özellik seçimi üzerine odaklanılabilir.

Sonuç olarak, Naive Bayes modelinin genel bir performansı iyi olsa da, yanlış sınıflandırmaların sayısını azaltmak ve böylece modelin güvenilirliğini artırmak için iyileştirmeler yapılabilir.

Referanslar

[1] Purba, K. R., Asirvatham, D., & Murugesan, R. K. (2020). Classification of Instagram fake users using supervised machine learning algorithms. *International Journal of Electrical and Computer Engineering (IJECE)*, 10(3), 2763-2772.

[2] Baykal, A. (2006). Veri Madenciliği Uygulama Alanları. *D.Ü. Ziya Gökalp Eğitim Fakültesi Dergisi*, 7, 95-107.

[3] Ersöz, F., & Çınar, Y. (2021). Veri Madenciliği ve Makine Öğrenimi Yaklaşımlarının Karşılaştırılması: Tekstil Sektöründe bir Uygulama. *Avrupa Bilim ve Teknoloji Dergisi*, (29), 397-414.

[4] Pekel Özmen, E., & Özcan, T. (2019). Dolandırıcılık Tespiti Üzerine Melez Sınıflandırma Ve Regresyon Ağacı Uygulaması. *Yönetim Bilişim Sistemleri Dergisi*, 5(2), 12-20.

[5] Ari, A., & Berberler, M. E. (2017). Yapay Sinir Ağları ile Tahmin ve Sınıflandırma Problemlerinin Çözümü İçin Arayüz Tasarımı. *Acta Infologica*, 1(2), 55-73.

[6] Ercan, T., & Kutay, M. (2016). Endüstride Nesnelerin Interneti (IoT) Uygulamaları. *Afyon Kocatepe University Journal of Science and Engineering*, 16, 035102 (599-607).

[7] Gündüz, M. Z., & Daş, R. (2018). Nesnelerin interneti: Gelişimi, bileşenleri ve uygulama alanları. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 24(2), 327-335.

[8] Sazak, T., Albayrak, Y., Nesnelerin İnterneti (IoT) Üzerine Ortam Verilerini Toplayan ve Uzaktan Takibini Sağlayan Bir Sistem Tasarımı, Akdeniz Üniversitesi, Elektrik Elektronik Mühendisliği Bölümü, Antalya

[9] Yiğitbaşı, Z. H. (2011). Nesnelerin İnterneti Ve Makineden Makineye Kavramları İçin Kilit Öncül - IPv6. *Ulusal IPv6 Konferansı*.

[10] Rapidminer. (n.d.). Naive Bayes. Erişim tarihi: [Aralık, 2023], https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/bayesian/naive_bayes.html.

[11] Rapidminer. (n.d.). Parallel Decision Tree. Erişim tarihi: [Aralık, 2023], https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/parallel_decision_tree.html

[12] Rapidminer. (n.d.). Deep Learning. Erişim tarihi: [Aralık, 2023], https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/neural_nets/deep_learning.html

[13] Rapidminer. (n.d.). Bagging. Erişim tarihi: [Aralık, 2023], https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/ensembles/bagging.html)